Getting Started with Text Analysis

Learning Outcomes

By the end of this topic, you will have achieved the following learning outcomes:

- I can understand the concept of text analysis in data science.
- I can apply text processing techniques to data in preparation for further analysis.
- I can distinguish between different text analysis techniques.

"If you want to understand people, especially your customers, then you have to be able to possess a strong capability to analyze text." Paul Hofmann

Reading

Overview

In data science, we normally work with **structured** and **unstructured data**. **Structured data** is highly organized and can be stored in databases while on the other hand, **unstructured data,** which will be our main focus in the session, is unorganized data that may or may not fit well in a database.

Unstructured data can be textual and non-textual i.e. media files (audio and video files, images), text files (word docs, powerPoint presentations), communication data (chat, call recordings), etc.

This session will help us get started with text analysis which entails working with unstructured data. This will serve as an introduction to **Natural language processing (NLP)** which is a branch of artificial intelligence that deals with the interpretation and manipulation of human language. More specifically, we will work with textual data.

The objective will be to get a grasp of handling text fundamentals before undertaking higher level NLP capabilities such as:

- **Content Categorization/ Text Classification**
    - Classifying text into different categories.
- **Topic Discovery and Modeling**
    - Accurately capture the meaning and themes in text collections, and apply advanced analytics to text, like optimization and forecasting.
- **Contextual Extraction**
    - Automatically pull structured information from text-based sources.
- **Sentiment Analysis**
    - Identifying the mood or subjective opinions within large amounts of text, including average sentiment and opinion mining.
- **Document Summarization**
    - Automatically generating synopses of large bodies of text.
- **Speech-to-text and Text-to-speech Conversion**
    - Transforming voice commands into written text, and vice versa.
- **Machine translation**
    - Automatic translation of text or speech from one language to another.

What is Text Analysis?

Text analysis is the automated process of understanding and sorting unstructured text, making it easier to manage. Text analysis tools are often used to unearth valuable insights from social media conversations, survey responses, online reviews, and more.

Text analysis can be applied to:

- **Whole documents**
    - By obtaining information from a complete document or paragraph e.g. the overall sentiment of a customer review.
- **Single sentences**
    - By obtaining information from specific sentences e.g. more detailed sentiments of every sentence of a customer review.
- **Sub-sentences**
    - By obtaining information from sub-expressions within a sentence e.g. the underlying sentiments of every opinion unit of a customer review.

Use Cases and Applications

- Social Media Monitoring
- Brand Monitoring

- Customer Service
- Voice of Customer & Customer Feedback
- Business Intelligence
- Sales and Marketing
- Product Analytics
- Knowledge Management

Text Analysis Techniques

Below are some of the basic text processing techniques that we will come across while performing text analysis. We first perform text cleaning techniques in order to prepare data for analysis then later can perform other processing techniques in order to create new features.

Text Cleaning / Text Processing Techniques

a. **Frequency of Words**
   - We might want to determine the frequency of words from textual data.
   - We can calculate word frequency with filtering options such as words starting, containing or ending in a particular way or list of nouns, verbs and other parts of speech.
   - Determining the frequency of words helps us perform multiple words removal. It can also help us create a new feature in our dataset.

b. **Characters & Special Characters**
   - We can calculate the number of characters in textual data. This is done by calculating the length of the string or word, thus can be used when performing feature construction.
   - On the other hand, we can resolve to remove special characters i.e. removal of irrelevant numbers or even retrieve them in cases where they have some meaning such as when analyzing tweets, hashtags "#" might help us with the no. of mentions.

c. **Frequency of Numerics**
   - The frequency of numerical characters is also important in the case where numerical characters within the text are important.
   - Determining this would also be important in feature construction.

d. **Dealing with Stopwords**
   - Stopwords are words which do not add much meaning to a sentence such that they can safely be ignored without sacrificing the meaning of the sentence i.e as the, is, at, which, and on.

- Stopwords are not useful in model creation as they provide little to no unique information that can be used i.e. Language Classification, Spam Filtering, Caption Generation, Auto-Tag Generation, Sentiment analysis, or anything related to Text Classification.
- On the other hand, stop words can be useful in language translation because they have to be translated along with other words thus need to be retained in situations of Machine Translation, Question-Answering problems, Text Summarization, Language Modeling.
- The frequency of stopwords in text can also be useful in the creation of a new feature.

e. Lowercasing and Frequency of Uppercase Words

- We perform lowercasing to transform our text to lowercase which helps us to avoid multiple copies of the same words.
- In some other cases, we might want to identify the number of uppercase words as they could have an associated meaning such as rage or anger.
- The frequency of lowercase, uppercase and title case words can be used in feature construction.

f. Average Word Length

- The average word length is calculated while performing feature engineering i.e. We could create a new important feature such as average word length *(sum of the length of all words divided by the total length of text)*, in an effort to improve our model.

g. Punctuation

- Punctuation characters can be removed from textual data since they don't add any extra meaning to data.
- The no. of punctuation characters can be used in feature construction.

h. Rare words

- In some cases, we might be required to remove rarely occurring words from textual data. This action would be taken because their presence is associated with noise/outliers.
- Alternatively, rare words can be replaced with a more general form of words.

i. Spelling Correction

- Spelling correction allows spelling mistakes within text data to be rectified.
- This is a resource intensive process and is applicable to texts in the same language and the used library i.e. one would perform spelling correction on a english text with an library of english words, french text with a library of french words etc.

j. **Other basic data cleaning** / **pre-processing techniques** include:
   - Removal of emojis
   - Removal of emoticons
   - Conversion of emoticons to words
   - Conversion of emojis to words
   - Removal of URLs
   - Removal of HTML tags

Advanced Text Processing Techniques

While preparing our data for further analysis or modelling, we can perform some of the following techniques to ensure that our data is in the right shape.

a. **Tokenization**
   We can tokenize our data, which refers to splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms with each of these smaller units called tokens.

b. **Stemming**
   Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form. This would refer to the removal of suffices, like “ing”, “ly”, “s”, etc.

   - For example, if there are two words: walks and walking, then stemming will stem the suffix to make them: walk.
   - But say in another example where we have two words console and consoling, the stemmer will remove the suffix and make them consol which is not a proper English word. In such a case, the lemmatization technique is considered.

c. **Lemmatization**
   Lemmatization is similar to stemming in reducing inflected words to their word stem but differs in the way that it makes sure the root word (also called lemma) belongs to the language.
   - In the previous stemming example of consol, the root word would become console. Lemmatization is generally slower than the stemming process. So depending on the speed requirement, we can choose to use either stemming or lemmatization.

d. **N-grams**
   N-grams are a combination of multiple words used together. They are basically a set of co-occurring words within a given window and when computing the

n-grams you typically move one word forward (although you can move X words forward in more advanced scenarios).

- For example, for the sentence "The cow jumps over the fence".
    - If N=2 (known as bigrams), then the n-grams would be: the cow, cow jumps, jumps over, over the, the fence.
    - If N=3, the n-grams would be: the cow jumps, cow jumps over, jumps over the, over the fence.

N-grams are used for a variety of different tasks. For example, when developing a language model, n-grams are used to develop not just unigram models but also bigram and trigram models that can be used in a variety of tasks such as spelling correction, word breaking and text summarization.

We will use n-grams in the creation of TF-IDF word level or character level features.

e. Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a parameter that penalizes common words by assigning them lower weights while giving importance to less common words in a particular document.

This weight is a statistical measure (no.) used to evaluate how important a word is to a document in a collection. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

In order to prepare our text data for modeling through *Term Frequency-Inverse Document Frequency (TF-IDF)*, we can take our text through a **TfidfVectorizer** which will tokenize our text and create a sparse matrix containing numerical information. Thus making it ready to be taken in by an algorithm.

During the process of the creation of our new features, we can also choose whether the feature should be made of word or character n-grams. Alternatively, we could create two features, one made up of word n-grams while the other character n-grams.

f. Bag of Words

If we would like to have a sparse matrix, where the frequency of text content i.e. words, is important we can create a bag of words.

A bag of words is the simplest form of text representation in numbers. It allows us to represent a sentence as a bag of words (a string of numbers). Any information about the order or structure of words in the document is discarded.

The intuition is that documents are similar if they have similar content.

A bag of words sparse matrix can be created through the use of a *CountVectorizer*.

g. Word Embedding

Word Embedding is a language modelling technique used for mapping words to vectors of real numbers. Words or phrases are mapped to vectors of real numbers. This involves a mathematical embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension.

Word embeddings can be generated using various methods such as Word2Vec, fastText, etc.

- **Word2Vec**
  - Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words.
  - There are two flavours of word2vec, such as CBOW and Skip-Gram. Given a set of sentences (also called corpus), the model loops on the words of each sentence and either try to use the current word w in order to predict its neighbours (i.e., its context), this approach is called “Skip-Gram”, or it employs each of these contexts to predict the current word w, in that case, the approach is called “Continuous Bag Of Words” (CBOW).

Word2Vec as well as other word embeddings techniques will be covered more in depth upon going through neural networks.

As mentioned, we can create new features from our text resulting in meta features and sparse matrices:

1. **Text Based Features** (Resulting to Meta features)
   a. Number of words in the text
   b. Number of unique words in the text
   c. Number of characters in the text
   d. Number of stopwords
   e. Number of punctuations
   f. Number of uppercase words
   g. Number of title case words
   h. Average length of the words
2. **NLP Based Features** (Resulting to Meta features)
   a. Polarity
   b. Subjectivity

    c. Noun Count
    d. Verb Count
    e. Adjective Count
    f. Adverb Count
    g. Pronoun Count
3. **TF-IDF Features** (Resulting to Sparse Features)
    a. Word Level N-Gram TF-IDF
    b. Character Level N-Gram TF-IDF
4. **Word Embedding Features**
5. **Topic Modelling Features**
6. **Count Features**

Meta features and sparse features are later combined through stacking in order to create one final feature for use in modeling. Alternatively, we could build different models with sparse and meta features, then using ensemble learning combine their outcomes to get the final result.

Further Analysis

Upon performing the above text processing techniques, the outcome of our processing can undergo further analysis which would involve the following applications.

a. Text Classification

This is the process of assigning predefined tags or categories to unstructured text. It's considered one of the most useful Natural Language Processing (NLP) techniques because it's so versatile and can organize, structure and categorize pretty much anything to deliver meaningful data and solve problems.

Common text classification tasks include sentiment analysis, topic modelling, intent detection and language detection.

- **Sentiment Analysis**

  This is the automated process of understanding an opinion about a given subject from written or spoken language.

  For example, by using sentiment analysis companies are able to flag complaints or urgent requests, so they can be dealt with immediately – and perhaps avert a PR crisis on social media.

  Other uses of sentiment classifiers include assessing brand reputation, carrying out market research, and improving products with customer feedback.

- **Topic Modeling**
  Another common example of text classification is topic analysis/modelling or, more simply put, understanding what a given text is talking about. It's often used for structuring and organizing data.

  For example: “The app is really simple and easy to use”. This product feedback can be classified under “ease of use”.

- **Intent Detection**
  Text classifiers can also be used to automatically detect the intent within texts.

  For example, companies are able to better understand customer feedback about a product if they know more about the purpose or intentions behind the text. Anything from the intention to complain about a product to the intention to buy a product.

b. Text Extraction

Text extraction is another widely used text analysis technique for getting insights from data.

It involves extracting pieces of data that already exist within any given text, so if you wanted to extract important data such as keywords, prices, company names, and product specifications, you'd train an extraction model to automatically detect this information.

Text extraction is often used alongside text classification so that businesses can categorize their data and extract information at the same time. There are different extraction models for different types of purposes which includes keyword extraction, entity recognition,

c. Text Clustering

Text clustering involves the grouping of unstructured data.

Although less accurate than classification algorithms, clustering algorithms are faster to implement because you don't need to tag examples to train models. That means these smart algorithms mine information and make predictions without the use of training data, otherwise known as unsupervised machine learning.

d. Document Summarization

This would refer to shortening long pieces of text with the goal of creating a coherent and fluent summary having only the main points outlined in the document.

Text Analysis Tools

Python

- Python is the most widely used language in scientific computing, period. Tools like NumPy and SciPy have established it as a fast, dynamic language that calls C and Fortran libraries where performance is needed.
- These libraries, combined with a thriving community and a diverse set of libraries to implement NLP models has made Python one of the most preferred programming languages for doing text analysis.

NLTK

- NLTK, the Natural Language Toolkit, is a best-of-class library for text analysis tasks. There's plenty of code written with it and no shortage of users familiar with both the library and the theory of NLP who can help answer your questions within the community.

SpaCy

- SpaCy is an industrial-strength statistical NLP library. Aside from the usual features, it adds deep learning integration and convolutional neural network models for multiple languages. Unlike NLTK, which is a research library, SpaCy aims to be a battle-tested, production-grade library for text analysis.

Scikit-learn

- Scikit-learn is a complete and mature machine learning toolkit for Python built on top of NumPy, SciPy, and matplotlib, which gives it stellar performance and flexibility for building text analysis models.

References

You can also use the following resources for further reading.

1. What is Natural Language Processing? [Link]
2. Introduction to Text Mining [Link]
3. Dealing with Text Data [Link]
4. Extensive Text Data Feature Engineering [Link]